ECE 443 Lab 4

November 21, 2011

QPSK (Quadrature Phase-shift Keying)

This lab is actually quite similar to Lab 2. You may want to refer to that lab for techniques and Matlab code.

A QPSK (quadrature phase-shift keyed) signal can be written in the form

$$x(t) = \sum_{k=0}^{K} s[k]p(t - kT)$$

where $s[k]$ is a QPSK symbol stream and $p(t)$ is the truncated square root raised-cosine pulse

$$p(t) = \frac{4r}{\pi\sqrt{T}} \frac{\cos((1+r)\pi t/T) + \sin((1-r)\pi t/T)/(4rt/T)}{1-(4rt/T)^2}, \quad |t| \leq 3T.$$

You can set the symbol period $T = 4$ or 8 and approximate $x(t)$ with 4 or 8 samples for each symbol period. The rolloff factor r should be set between 0.25 and 0.5. For example if the sampling period is $T/4$,

$$x[n] = x(nT_s) = \sum_{k=0}^{K} s[k]p[n - 4k] = \sum_{k=0}^{K} (b_i[k] + jb_q[k])p[n - 4k].$$

Plot the pulse and its spectrum for several values of r and compare them. What does r do? Plot the spectrum of a flat pulse of width T and compare it to the spectrum of p. What is an advantage and disadvantage of p compared to a flat pulse? Finally, plot an example of the modulated signal x, with separate real and imaginary parts.

Tips:

- You can generate the QPSK data with

  $$\texttt{s} = \texttt{sign}(\texttt{randn}(1,\texttt{K})) + \texttt{j} * \texttt{sign}(\texttt{randn}(1,\texttt{K}))$$

  `randn`() generates normally distributed random numbers with a mean of 0 and variance of 1 (often written as $z \sim N(0,1)$). Normal variables have the mean equal to the median, so values above and below the mean are equally likely. Note that zero is a possibility, but is quite unlikely. If zero were to occur in either the real or imaginary part, we would have an invalid symbol (this is just how QPSK works). Think about how to avoid or fix zeros.

- You can do the sum by filtering the zero padded symbols

  $$u = [s[1], 0, 0, 0, s[2], 0, 0, 0, s[3], 0, \ldots]$$

  with $p[n]$ like so: $\texttt{x[n]} = \texttt{filter}(\texttt{p}, 1, \texttt{u})$. You must be careful of the time shift added by the `filter`() command. Alternatively, you can use the Kronecker product (`kron`()) as seen in Lab 2.

Demodulation

Demodulate $x[n]$ using a matched filter. This means filtering $x[n]$ with $p[n]$ and sampling every 4 (using the example sampling choice from above) points to obtain $z[k]$. This is where the timing from the `filter`() command may cause trouble. You can then detect the data with

$$\texttt{sign}(\texttt{real}(\texttt{z})) + \texttt{j} * \texttt{sign}(\texttt{imag}(\texttt{z}))$$

Since there is no noise, the error rate should be zero.

Noise

Now add noise $y[n] = x[n] + \sigma_N v[n]$, where $v[n] \sim N(0,1)$. Let the SNR to be $-20 \log \sigma_N$, after normalizing the signal power to 1. You can find the signal power with `norm(x)^2/length(x)`. Perform detection on the noisy data and calculate the bit error rate (BER) from 10^0 to 10^{-3}. You will have to start with a low SNR value and increase it with a step size of ≈ 0.5dB. Note that each symbol carries 2 bits of information (why?). Plot the BER vs. SNR. To have a reliable estimate of BER up to 10^{-3}, you need the size of the data to be on the order of 10^4.